🚫 Bloqueadores para Implementación - Epic 11

Fecha: 1 de diciembre de 2025
Proyecto: Signature Router - Integración MuleSoft
Estado: ❌ **NO READY - Bloqueado**

⚠️ RESUMEN EJECUTIVO

¿Se puede implementar con la información actual?

❌ NO

Razón:

Faltan **datos técnicos críticos** sin los cuales **no es posible hacer ni un solo request** a la API de MuleSoft.

Información Disponible:

- ✅ Documentación conceptual (~40%)
- ✅ Endpoints identificados
- ✅ Schema de ejemplo (SMS)
- ✅ Canales disponibles confirmados

Información Faltante:

- ❌ Credenciales de autenticación (~20%)
- ❌ URLs completas de ambientes (~15%)
- ❌ Schemas completos y validados (~10%)
- ❌ Manejo de errores (~10%)
- ❌ Rate limits y SLAs (~5%)

🔴 BLOQUEADORES CRÍTICOS

Sin resolver estos puntos, **es IMPOSIBLE empezar el desarrollo**.

---

1. 🔐 NO Tenemos Credenciales de Autenticación

Problema:
No podemos autenticarnos con la API de MuleSoft.

Lo que nos falta:

- ❌ Client ID
- ❌ Client Secret
- ❌ Método de autenticación (¿OAuth2? ¿API Key? ¿JWT?)
- ❌ Token endpoint (si es OAuth2)
- ❌ Scopes necesarios (si es OAuth2)
- ❌ Tiempo de expiración de tokens

Impacto:

```
# Esto NO funcionará sin credenciales:
curl -X POST https://api.selfbank.es/communication-execution/sms-
notification/execute \
  -H "Authorization: ???"  # ← NO sabemos qué poner aquí
  -d '{ ... }'

# Response esperado:
HTTP 401 Unauthorized
{
  "error": "Authentication required"
}
```

Sin esto:

- ❌ No podemos hacer **NINGÚN** request
- ❌ No podemos probar en DEV
- ❌ No podemos desarrollar
- ❌ No podemos hacer testing

Estado: 🔴 BLOQUEADOR ABSOLUTO

---

2. 🌐 NO Tenemos URLs Completas de Ambientes

Problema:
No sabemos a dónde enviar los requests.

Lo que tenemos:

```
✅ Endpoints relativos:
   /communication-execution/sms-notification/execute
   /communication-execution/push-notification/execute
```

Lo que nos falta:

```
❌ URL base completa de DEV:  https://???
❌ URL base completa de UAT:  https://???
❌ URL base completa de PROD: https://???
```

Pista parcial:
En el API Manager vimos: `https://api.selfbank.es/system/commu...` (truncada)

Posibilidades (sin confirmar):

```
¿Es https://api.selfbank.es/system/communication/v1 ?
¿O https://api-dev.selfbank.es/communication/v1 ?
¿O https://mulesoft.selfbank.es/api/v1 ?
```

Impacto:

```
# NO sabemos qué URL usar:
curl -X POST https://???/communication-execution/sms-notification/execute
                      ↑↑↑
                  ¿QUÉ VA AQUÍ?
```

Sin esto:

- ❌ No podemos hacer requests
- ❌ No podemos configurar el cliente HTTP
- ❌ No podemos hacer testing

Estado: 🔴 BLOQUEADOR ABSOLUTO

---

3. ❓ Campo "practice": "monkey" Sin Documentar

Problema:
Aparece en el ejemplo de request pero **NO está explicado**.

Ejemplo de la documentación:

```
{
    "customerId": "CUST12345678",
    "practice": "monkey",  // ← ¿¿¿QUÉ ES ESTO???
    "channel": "SMS",
    "recipient": { ... }
}
```

Lo que NO sabemos:

- ❌ ¿Qué significa "practice"?
- ❌ ¿Es obligatorio?
- ❌ ¿Qué valores acepta? ("monkey", ¿otros?)
- ❌ ¿Afecta al routing?
- ❌ ¿Afecta al proveedor usado?
- ❌ ¿Es un campo de testing?

Riesgos:

Escenario A: Si es obligatorio y no lo enviamos

```
Request sin "practice":
{
    "customerId": "CUST12345678",
    "channel": "SMS",
    ...
}

Response esperado:
HTTP 400 Bad Request
{
  "error": "Missing required field: practice"
}
```

Escenario B: Si "monkey" es un valor inválido

```
Request con "monkey":
{
    "practice": "monkey",  // ¿Es válido?
    ...
}

Response posible:
HTTP 400 Bad Request
{
  "error": "Invalid value for practice: monkey"
}
```

Escenario C: Si afecta al comportamiento

```
// ¿"monkey" envía a un provider de testing?
// ¿"production" envía a provider real?
// NO LO SABEMOS
```

Sin esto:

- ⚠️ Requests podrían fallar con 400 Bad Request
- ⚠️ Comportamiento impredecible
- ⚠️ Posible envío a proveedores incorrectos

Estado: 🔴 BLOQUEADOR CRÍTICO

---

4. 📋 NO Tenemos Schema Completo de PUSH Notifications

Problema:
Solo tenemos schema detallado de SMS, PUSH está incompleto.

Lo que tenemos (SMS):

```
{
    "customerId": "CUST12345678",
    "channel": "SMS",
    "recipient": {
        "phoneNumber": "+34653093774",
        "countryCode": "ES"
    },
    "content": {
        "message": "Texto del mensaje",
        "encoding": "UTF8"
    },
    "smsOptions": {
        "senderId": "SELFBANK",
```

```
        "validityPeriod": 60,
        "deliveryReport": true
    }
}
```

Lo que NO tenemos (PUSH):

```
{
    "customerId": "CUST12345678",
    "channel": "PUSH",
    "recipient": {
        "deviceToken": "???"  // ← ¿Cómo se especifica?
        // ¿deviceId?
        // ¿registrationToken?
        // ¿Qué formato?
    },
    "content": {
        "title": "???",
        "body": "???",
        "data": { ??? }  // ← ¿Qué estructura?
    },
    "pushOptions": {
        // ← ¿Qué opciones hay?
        // ¿priority?
        // ¿badge?
        // ¿sound?
        // ¿timeToLive?
    }
}
```

Preguntas sin respuesta:

- ❌ ¿Cómo se especifica el deviceToken?
- ❌ ¿Formato de payload?
- ❌ ¿Diferencia entre iOS y Android?
- ❌ ¿Soporte para rich notifications?
- ❌ ¿Opciones de prioridad?
- ❌ ¿Campos obligatorios vs opcionales?

Sin esto:

- ❌ **NO podemos implementar PUSH**
- ⚠️ Solo podríamos implementar SMS

Estado: 🔴 **BLOQUEADOR para PUSH** (SMS podría funcionar)

🟡 BLOQUEADORES IMPORTANTES

Estos NO impiden empezar, pero **dificultan una implementación robusta y production-ready**.

---

5. 📊 NO Sabemos Rate Limits ni Timeouts

Problema:
No conocemos las limitaciones de la API.

Lo que NO sabemos:

- ❌ ¿Cuántos requests por segundo podemos hacer?
- ❌ ¿Es por IP? ¿Por Client ID? ¿Por aplicación?
- ❌ ¿Qué timeout recomiendan?
- ❌ ¿Qué pasa si excedemos el límite?
- ❌ ¿Hay `Retry-After` header en 429?

Riesgo A: Rate Limiting sin conocer límite

```
# Si enviamos demasiados requests:
for i in {1..1000}; do
  curl -X POST https://api/sms/execute ...
done

# Posible respuesta:
HTTP 429 Too Many Requests
{
  "error": "Rate limit exceeded: 100 requests per minute"
}

# Resultado: Sistema bloqueado, no podemos enviar SMS
```

Riesgo B: Timeout mal configurado

```
// Si configuramos timeout muy bajo:
HttpClient.builder()
    .timeout(Duration.ofSeconds(1))  // ¿Es suficiente?
    .build();

// Si MuleSoft tarda 2 segundos:
// → TimeoutException
// → SMS NO se envía aunque la operación fue exitosa
// → Cliente NO recibe código
// → Proceso de firma fallido
```

Riesgo C: Retry sin estrategia

```
// Sin conocer la retry policy recomendada:
// Podríamos causar "retry storm"
for (int i = 0; i < 5; i++) {
    try {
        sendSMS();
        break;
    } catch (Exception e) {
        Thread.sleep(100);  // ¿Backoff correcto?
        // ¿Reintentamos en 429?
        // ¿Reintentamos en 500?
        // ¿Reintentamos en 503?
    }
}
```

Sin esto:

- ⚠️ Posible bloqueo de IP/Client por exceder rate limits
- ⚠️ Timeouts incorrectos (falsos negativos o lentitud)
- ⚠️ Retry storms (empeorar el problema)
- ⚠️ No podemos configurar circuit breaker correctamente

Estado: 🟡 **BLOQUEADOR IMPORTANTE** (se puede empezar con valores conservadores)

---

6. ❌ NO Tenemos Documentación de Errores

Problema:
Solo vimos response exitoso (200 OK), no sabemos cómo son los errores.

Lo que tenemos:

```
// Response 200 OK:
{
    "notificationId": "COMM-EXEC-20241209-001234",
    "status": "SENT",
    "submittedAt": "2024-12-09T15:30:25.123Z"
}
```

Lo que NO tenemos:

```
// ¿Cómo es un 400 Bad Request?
{
    "error": "???" ,
    "code": "???",
    "message": "???",
    "details": { ??? }
}

// ¿Cómo es un 401 Unauthorized?
// ¿Cómo es un 429 Too Many Requests?
// ¿Cómo es un 500 Internal Server Error?
// ¿Cómo es un 503 Service Unavailable?
```

Preguntas sin respuesta:

- ❌ ¿Qué códigos de error existen?
- ❌ ¿Formato del error response?
- ❌ ¿Códigos de error específicos? (INVALID_PHONE, PROVIDER_DOWN, etc.)
- ❌ ¿Cómo distinguir error de validación vs error de provider?

Impacto en el código:

```
// Manejo de errores genérico (pobre):
try {
    Response response = httpClient.post(request);
    if (response.status() != 200) {
        throw new RuntimeException("Error: " + response.status());
        // ↑ Mensaje genérico, no ayuda al troubleshooting
    }
} catch (Exception e) {
    // ¿Qué hacemos aquí?
    // ¿Reintentamos?
    // ¿Fallback a otro canal?
    // ¿Notificamos al usuario?
    // NO LO SABEMOS
}
```

Casos problemáticos:

Caso 1: Número de teléfono inválido

```
// Request con número inválido:
{
    "recipient": {
        "phoneNumber": "123"  // ← Inválido
    }
```

```
}

// ¿Respuesta esperada?
HTTP 400
{
    "error": "INVALID_PHONE_NUMBER"  // ← ¿Así?
}

// ¿O simplemente?
HTTP 400
{
    "message": "Bad request"  // ← No ayuda
}
```

Caso 2: Provider caído

```
// Si Twilio está DOWN:

// ¿Respuesta esperada?
HTTP 503 Service Unavailable
{
    "error": "PROVIDER_UNAVAILABLE"
}

// ¿O MuleSoft intenta con fallback y devuelve 200?
// NO LO SABEMOS
```

Sin esto:

- ⚠️ Manejo de errores pobre (mensajes genéricos)
- ⚠️ Troubleshooting muy difícil
- ⚠️ No podemos diferenciar errores recuperables vs no recuperables
- ⚠️ Experiencia de usuario degradada

Estado: 🟡 **BLOQUEADOR IMPORTANTE** (se puede empezar con try-catch genérico)

---

7. 🔍 NO Sabemos Qué Providers Usan

Problema:
No sabemos qué proveedores externos usa MuleSoft.

Lo que NO sabemos:

- ❌ SMS: ¿Twilio? ¿Nexmo? ¿Vonage? ¿AWS SNS?
- ❌ PUSH: ¿Firebase FCM? ¿OneSignal? ¿AWS SNS?

- ❌ ¿Tienen múltiples providers por canal?
- ❌ ¿Hay fallback automático?

Impacto A: No podemos estimar latencias

```
Twilio típico:     50-150ms
Nexmo típico:      80-200ms
AWS SNS típico:    30-100ms

Sin saber cuál usan:
→ No podemos prometer SLAs precisos a clientes
→ No podemos configurar timeouts óptimos
```

Impacto B: No podemos estimar costos

```
Twilio US:  $0.0075 por SMS
Nexmo US:   $0.0064 por SMS
AWS SNS:    $0.00645 por SMS

Diferencia potencial: ~17% en costos

Sin saber cuál usan:
→ No podemos hacer proyección de costos
→ No podemos optimizar presupuesto
```

Impacto C: No podemos anticipar limitaciones

```
Twilio:
  - Límite de 1600 caracteres por SMS
  - Soporte para emojis: Sí
  - Entrega internacional: 200+ países

Nexmo:
  - Límite de 3200 caracteres por SMS
  - Soporte para emojis: Limitado
  - Entrega internacional: 190+ países

Sin saber cuál usan:
→ No sabemos las limitaciones reales
→ Posibles fallos inesperados
```

Sin esto:

- ⚠️ No podemos optimizar performance
- ⚠️ No podemos hacer proyección de costos

- ⚠️ No podemos anticipar limitaciones
- ⚠️ Epic 9 (Analytics Dashboard) muy limitado

Estado: 🟡 **BLOQUEADOR para Analytics** (SMS/PUSH funcionarían pero sin visibilidad)

📊 MATRIZ DE BLOQUEADORES

#	Bloqueador	Severidad	¿Impide Empezar?	¿Impide Testing?	¿Impide Producción?
1	**Credenciales**	🔴 CRÍTICO	✅ SÍ	✅ SÍ	✅ SÍ
2	**URLs completas**	🔴 CRÍTICO	✅ SÍ	✅ SÍ	✅ SÍ
3	**Campo "practice"**	🔴 CRÍTICO	⚠️ Parcial	⚠️ Parcial	✅ SÍ
4	**Schema PUSH completo**	🔴 CRÍTICO	⚠️ Solo PUSH	⚠️ Solo PUSH	⚠️ Solo PUSH
5	**Rate limits**	🟡 IMPORTANTE	❌ NO	❌ NO	⚠️ Riesgoso
6	**Documentación errores**	🟡 IMPORTANTE	❌ NO	❌ NO	⚠️ Riesgoso
7	**Providers reales**	🟡 IMPORTANTE	❌ NO	❌ NO	⚠️ Sin analytics

---

🎯 ESCENARIOS DE IMPLEMENTACIÓN

Escenario A: Con TODA la información faltante

```
✅ Credenciales: SÍ
✅ URLs: SÍ
✅ "practice" explicado: SÍ
✅ Schema PUSH completo: SÍ
✅ Rate limits: SÍ
✅ Docs de errores: SÍ
✅ Providers conocidos: SÍ

Resultado:
→ ✅ Implementación completa (SMS + PUSH)
→ ✅ Production-ready
```

```
→ ✅ Con analytics (Epic 9)
→ ⏱️ Timeline: 3-4 sprints
```

Escenario B: Solo con MÍNIMO (después de reunión lunes)

```
✅ Credenciales: SÍ
✅ URLs: SÍ
✅ "practice" explicado: SÍ
✅ Schema PUSH completo: SÍ
❌ Rate limits: NO → Usar valores conservadores
❌ Docs de errores: NO → Try-catch genérico
❌ Providers conocidos: NO → Sin analytics detallado

Resultado:
→ ✅ Implementación funcional (SMS + PUSH)
→ ⚠️ Production-ready con limitaciones
→ ❌ Sin analytics detallado
→ ⏱️ Timeline: 3-4 sprints (mismo tiempo, menos calidad)
```

Escenario C: Situación ACTUAL (sin reunión)

```
❌ Credenciales: NO
❌ URLs: NO
❌ "practice" explicado: NO
❌ Schema PUSH completo: NO
❌ Rate limits: NO
❌ Docs de errores: NO
❌ Providers conocidos: NO

Resultado:
→ ❌ NO se puede implementar NADA
→ ❌ NO se puede hacer testing
→ ❌ Desarrollo bloqueado
→ ⏱️ Timeline: Indefinido (bloqueado hasta obtener info)
```

🚦 SEMÁFORO DE READY-STATE

Estado Actual: 🔴 RED - Bloqueado

```
Información disponible:  ████████░░░░░░░░░░░░ 40%
Información necesaria:   ████████████████████ 100%
Gap crítico:             ████████████░░░░░░░░ 60%

Estado: BLOQUEADO
```

Después de Reunión Lunes (esperado): 🟡 YELLOW - Puede empezar con limitaciones

```
Información disponible:  ████████████████░░░░ 80%
Información necesaria:   ████████████████████ 100%
Gap crítico:             ████░░░░░░░░░░░░░░░░ 20%

Estado: PUEDE EMPEZAR (con limitaciones)
```

Con Toda la Información: 🟢 GREEN - Production Ready

```
Información disponible:  ████████████████████ 100%
Información necesaria:   ████████████████████ 100%
Gap crítico:             ░░░░░░░░░░░░░░░░░░░░ 0%

Estado: PRODUCTION READY
```

📋 CHECKLIST DE DEFINITION OF READY

Para EMPEZAR desarrollo (MÍNIMO):

- ◯ **Credenciales de DEV** (Client ID + Secret)
- ◯ **URL de DEV completa** (https://???)
- ◯ **Método de autenticación** (OAuth2, API Key, etc.)
- ◯ **Explicación del campo "practice"** (qué es, valores válidos)
- ◯ **Schema completo de SMS** (ya lo tenemos, validar)
- ◯ **Schema completo de PUSH** (deviceToken, payload, opciones)

Sin estos 6 puntos: ❌ NO se puede empezar

Para TESTING robusto:

- ◯ Todo lo anterior +
- ◯ **Documentación de errores** (códigos, formatos)
- ◯ **Datos de prueba válidos** (teléfonos, deviceTokens)
- ◯ **Comportamiento esperado de errores** (400, 401, 429, 500, 503)

Para ir a PRODUCCIÓN:

- ◯ Todo lo anterior +
- ◯ **Credenciales de PROD** (Client ID + Secret)
- ◯ **URL de PROD completa**

- ◯ **Rate limits confirmados** (requests/seg)
- ◯ **Timeouts recomendados**
- ◯ **Retry policy recomendada**
- ◯ **SLAs de disponibilidad** (99.9%?)
- ◯ **Proceso de escalación** (contacto de soporte)

Para ANALYTICS completo (Epic 9):

- ◯ Todo lo anterior +
- ◯ **Providers reales conocidos** (Twilio, Firebase, etc.)
- ◯ **Metadata en responses** (provider, latencia, coste)
- ◯ **Especificación de /health endpoint**
- ◯ **Especificación de /metrics endpoint**

⏱️ TIMELINE ESTIMADO

Situación Actual → Reunión Lunes:

```
📅 Hoy (1 dic): 🔴 Bloqueado
📅 Lunes (reunión): Obtener MÍNIMO necesario
📅 Lunes tarde: 🟡 Puede empezar (si obtenemos credenciales)
```

Desarrollo → Producción:

```
📅 Semana 1-2: Setup inicial + primer request de prueba
📅 Semana 3-4: Implementación SMS
📅 Semana 5-6: Implementación PUSH
📅 Semana 7-8: Testing integración
📅 Semana 9-10: UAT
📅 Semana 11-12: Go-live

Total: ~3 meses (12 semanas) desde que obtengas MÍNIMO necesario
```

🎯 CONCLUSIÓN

¿Por qué NO se puede implementar ahora?

Respuesta simple:

> *No tenemos las "llaves" para acceder a la API (credenciales) ni sabemos "dónde está la puerta" (URLs).*

Analogía:

```
Es como tener el manual de un coche (documentación) pero:
❌ No tenemos las llaves (credenciales)
❌ No sabemos dónde está aparcado (URLs)
❌ No sabemos cómo arrancar (método de autenticación)
❌ Hay un botón misterioso sin explicar (campo "practice")

Resultado: No podemos conducirlo
```

¿Qué necesitamos URGENTEMENTE?

TOP 4 CRÍTICOS (sin esto, NADA funciona):

1. 🔐 Credenciales de DEV
2. 🌐 URL de DEV completa
3. ❓ Explicación de "practice": "monkey"
4. 📋 Schema completo de PUSH

¿Cuándo podemos empezar?

Después de la reunión del lunes, si obtenemos el TOP 4 crítico.

Tiempo estimado:

- ⏱️ Lunes reunión → Obtener info
- ⏱️ Lunes tarde → Primer request de prueba
- ⏱️ Martes → Setup completo
- ⏱️ Miércoles → Iniciar desarrollo real

Documento creado: 1 de diciembre de 2025
Próxima actualización: Después de reunión del lunes
Estado: 🔴 **BLOQUEADO - Esperando información crítica**